MISB Motion Imagery Standards Board **STANDARD** **Real-Time Protocol for Motion Imagery and Metadata**	**MISB ST 0804.4** **27 February 2014**

1 Scope

This Standard (ST) documents the guidelines for packaging and streaming motion imagery and metadata when using the Real-Time Protocol (RTP).

The scope of this document is limited to delivery of motion imagery products, and is not intended to replace any other approved standards for other uses; rather it is intended to complement those standards.

2 References

2.1 Normative References

The following references and the references contained therein are normative.

[1] ISMA 2.0, Internet Streaming Media Alliance Implementation Specification, Jul 2005
[2] IETF RFC 3550, RTP: A Transport Protocol for Real-Time Applications, Jul 2003
[3] IETF RFC 3551, RTP Profile for Audio and Video Conferences with Minimal Control, Jul 2003
[4] ISO/IEC 13818-2: 2000, Information Technology – Generic coding of moving pictures and associated audio information: Video
[5] ITU-T Rec. H.264 (04/2013), Advanced Video Coding for Generic Audiovisual Services, (ISO/IEC 14496-10:2012)
[6] MISB RP 0802.2 H.264/AVC Motion Imagery Coding, Feb 2014
[7] SMPTE ST 336:2007, Data Encoding Protocol Using Key-Length-Value
[8] IETF RFC 6597, RTP Payload Format for SMPTE ST 336 Encoded Data, Apr 2012
[9] IETF RFC 2250, RTP Payload Format for MPEG1/MPEG2 Video, Jan 1998
[10] DVB IP Phase 1 handbook, ETSI TS 102 034, Digital Video Broadcasting (DVB); Transport of MPEG-2 Based DVB Services over IP Based Networks, Mar 2005
[11] IETF RFC 6184, RTP Payload Format for H.264 Video, May 2011
[12] SMPTE 2022-2:2007, Unidirectional Transport of Constant Bit Rate MPEG-2 Transport Streams on IP Networks
[13] Pro-MPEG Code of Practice #3 release 2, Transmission of Professional MPEG-2 Transport Streams over IP Networks, Jul 2004
[14] MISB ST 0604.3 Time Stamping Compressed Motion Imagery, Feb 2014

[15] MISB ST 1402 MPEG-2 Transport of Compressed Motion Imagery and Metadata, Feb 2014
[16] IETF RFC 2326, Real Time Streaming Protocol (RTSP), Apr 1998
[17] IETF RFC 4566, SDP: Session Description Protocol, Jul 2006
[18] IETF RFC 768, User Datagram Protocol, Aug 1980

2.2 Informative References

[19] IETF RFC 2974, Session Announcement Protocol, Oct 2000
[20] IP Streaming of MPEG-4: Native RTP vs. MPEG-2 Transport Stream, Envivio, Oct 2005
[21] RTP - Audio and Video for the Internet, Colin Perkins, Nov 2006 ISBN-10: 0321833627
[22] SMPTE RP 217:2001, Non-Synchronized Mapping of KLV Packets into MPEG-2 Transport Streams

3 Acronyms

AVC	Advanced Video Coding
DTS	Decode Time Stamp
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISMA	Internet Streaming Media Alliance
NAT	Network Address Translation
PTS	Presentation Time Stamp
RTP	Real Time Protocol
RTP/AVP	RTP using Audio/Video profile carried over UDP
RTCP	Real Time Control Protocol
RTSP	Real Time Streaming Protocol
SAP	Session Announcement Protocol
SDP	Session Description Protocol
TCP	Transmission Control Protocol
TS	MPEG-2 Transport Stream
UDP	User Datagram Protocol
URL	Uniform Resource Locator

4 Revision History

Revision	**Date**	**Summary of Changes**
0804.4	12/18/2013	• Promoted to Standard • Revised to conform to EARS for requirements • Updated References

5 Introduction

This document defines the requirements for transporting motion imagery and metadata when using Real Time Protocol (RTP) [2]. A RTP transport provides for carriage of one type of media stream; this can be a motion imagery, metadata, or an audio elementary stream; it can also be a MPEG-2 Transport Stream that may contain a mix of these elementary streams. Guidance for both modes is provided. In addition, the de multiplex and mapping of MPEG-2 Transport Stream components to individual RTP streams is discussed. The scope of what this document is attempting to provide is very broad, and given the wide variety of infrastructure, client device, user requirements, and other considerations, it does not attempt to address all possible configurations. Implementation information for applying RTP is provided.

Unlike MPEG-2 Transport Stream (TS), RTP does not support the multiplexing of media streams together within one unified container; each media stream is carried as a separate RTP stream. As such, each RTP media stream must contain sufficient timing information for synchronizing related streams at the receiver. Mapping a motion imagery elementary stream, such as a compressed MPEG-2 or H.264/AVC elementary stream directly to RTP is called *native RTP carriage* and is defined in a *payload format* that specifies how that media type is to be transported over RTP. MPEG-2 Transport Stream (TS) can also be carried over RTP, which requires additional header overhead and is also defined by a payload format. Because both native and TS payload formats are common, they are included here for completeness.

Appendix A outlines a payload format for streaming KLV-encoded metadata over RTP. Example software code is available for download from the MISB web site.

5.1 RTP Motion Imagery Features

RTP is designed to accommodate the nuances of internet-centric multimedia streaming. It offers the following capabilities:

- Delivery of digital motion imagery over various network and link types that may exhibit packet loss, packet re-ordering, latency, and jitter.
- A standardized method for requesting an RTP stream from a digital motion imagery provider.
- A standardized method to allow trick play stream control.
- Authentication and encryption of data for integrity, confidentiality and non-repudiation.
- Provisions for lowering the overhead associated with packetizing and streaming data.

5.2 Relationship with established Internet Standards

The Internet provides a good example of the challenges in delivering data over disparate best-effort networks of varying qualities. ISMA [1] defines a set of standards for storage and streaming of media over the Internet; this document aligns itself with that family of standards.

5.3 Relationship with established MISB Standards

Figure 1 depicts the relationships between the current MISP standards and those in this document.

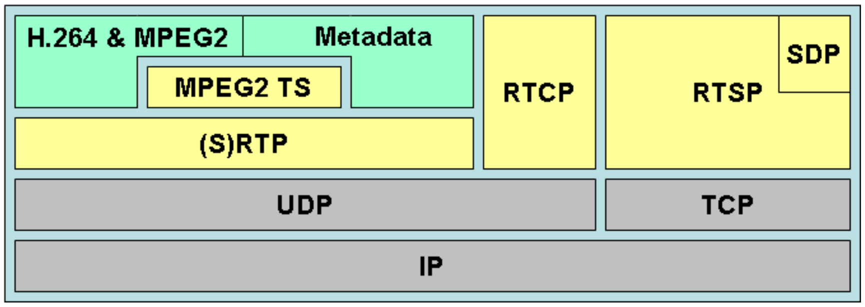

Figure 1: Relationship of RTP with MISP Standards

5.4 Transport

The Real-time Transport Protocol (RTP) along with its associated profiles and payload formats is a key standard for video/audio transport over IP networks. RTP provides end-to-end transport for media with real-time characteristics and is widely used in Internet streaming media applications. The IETF specifies RTP in RFC 3350 [2]. In applying RTP in system design RTP needs to be profiled. A profile is accompanied by several payload format specifications that describe how to transport a particular media format over RTP. The IETF initially published RFC 3551[3] with RTP, which is a profile describing video and audio conference applications with minimal control. This profile is referred as RTP/AVP [3].

Requirement	
ST 0804.4-01	Systems delivering Motion Imagery over RTP shall implement IETF RFC 768[18].
ST 0804.4-02	Systems delivering Motion Imagery over RTP shall implement IETF RFC 3550[2].
ST 0804.4-03	Systems delivering Motion Imagery over RTP shall implement IETF RFC 3551[3].

Rationale: Most current internet streaming media systems support RTP/AVP/UDP profile because it is simple and widely adopted.

Interleaved RTSP and RTP/AVP over TCP is an OPTIONAL method for transport. This method offers reliable transmission, and more easily traverses Network Address Translation (NAT) devices and Firewalls at the expense of real-time time-critical response.

6 Data Formats

This section provides guidance on the media formats that a RTP client must decode and display.

Requirement	
ST 0804.4-04	A RTP client shall support both motion imagery and metadata.

Optionally, the RTP client may support decoding and playing of audio.

6.1 Motion Imagery Format

The motion imagery compression types allowed are MPEG-2 [4] and H.264/AVC [5]. Coding parameters for H.264/AVC are defined in MISB RP 0802 [6].

Requirement	
ST 0804.4-05	When compressing motion imagery using H.264/AVC, a RTP server shall support H.264/AVC compression in accordance with ITU-T Rec. H.264 [5] and MISB ST 0604 [14].
ST 0804.4-06	When compressing motion imagery using H.264/AVC, a RTP server shall encode compressed motion imagery in accordance with MISB RP 0802 [6].
ST 0804.4-07	When compressing motion imagery using MPEG-2, a RTP server shall support ISO/IEC 13818-2 compression in accordance with ISO/IEC 13818-2 [4].
ST 0804.4-08	When receiving H.264/AVC compressed motion imagery, a RTP client shall support decoding and displaying motion imagery in accordance with ITU-T Rec. H.264 [5].
ST 0804.4-09	When receiving MPEG 2 compressed motion imagery a RTP client shall support decoding and displaying motion imagery in accordance with ISO/IEC 13818-2[4].

6.2 Metadata Format

The allowed metadata media format to be transmitted over RTP is limited to KLV-encoded data as specified by SMPTE ST 336 [7], and whose payload format is defined by IETF RFC 6597 [8].

Requirement	
ST 0804.4-10	A RTP server shall support encoded KLV-metadata in accordance with IETF RFC 6597 [8].
ST 0804.4-11	A RTP client shall support decoding and displaying metadata in accordance with IETF RFC 6597 [8].

7 RTP Payload Format for MPEG-2 and H.264

The RTP header provides a data field for a media stream timestamp. When referenced to a common wall clock, this local timestamp provides a means to synchronize multiple RTP streams at a decoder.

The H.264/AVC payload format over RTP is specified in IETF RFC 6184 [11]; the MPEG-2 payload format over RTP is specified by the IETF RFC 2250 [9] and by the DVB-IPI [10]. Additional parameter guidance is found in MISB RP 0802 [6].

Requirement	
ST 0804.4-12	The Single NAL unit mode and Non-interleaved mode packetization modes (packetization-mode=0, 1) shall be supported (guarantees lower latency) in accordance with IETF RFC 6184 [8].
ST 0804.4-13	Only the Single NAL unit mode and Non-interleaved mode packetization modes (packetization-mode=0, 1) shall be allowed in accordance with IETF RFC 6184 [8].
ST 0804.4-14	All parameters except those defined for the interleaved mode (packetization-mode=2) shall be allowed in the format parameters line (“a=fmtp”) in the SDP.
ST 0804.4-15	All parameters except max-mbps, max-fs, max-dpb, max-br shall be allowed in the format parameters line (“a = fmtp”) of the SDP. (These parameters extend the capabilities of a particular level specified by profile-level-id, but not all receivers may be able to decode streams beyond the profile-level-id specified)

ST 0804.4-16	The format parameters line ("a=fmtp") in the SDP shall include the parameters sprop-parameters-sets and profile-level-id.
ST 0804.4-17	When the SDP data within a stream changes, the stream shall be restarted after the new SDP data is communicated to the client.

It is recommended that the Sequence Parameter Set and Picture Parameter Set defined in [5] be carried in-band within the elementary stream. This should convey the same information as represented by sprop-parameter-sets.

8 RTP Payload Format for KLV-encoded Metadata

This document provides guidelines for RTP carriage of metadata that is consistent with MISB-approved metadata sets encoded into KLV (Key-Length-Value) according to SMPTE ST 336 [7]. IETF RFC 6597 [8] provides payload format guidance for metadata encoded according to SMPTE ST 336 over RTP.

9 RTP Payload Format for MPEG-2 Transport Stream

MPEG-2 Transport Stream over RTP incurs additional overhead and increases stream bit rate. There are benefits in transporting MPEG-2 TS over RTP. The packet count and time stamp in an RTP header provides a receiver the ability to detect lost and out-of-order packets, and in quantifying the jitter in packets received. MPEG-2 Transport Stream over RTP may be useful to transition from legacy tools, or to connect between points where such protocols may be difficult to modify.

Requirement	
ST 0804.4-18	Encapsulating MPEG-2 Transport Stream in RTP shall be in accordance with IETF RFC 2250 [9] with further restrictions on IETF RFC3550 [2] and RFC2250 specified in SMPTE 2022-2[12] and Pro-MPEG Code of Practice #3[13].

10 Native RTP Streams from MPEG-2 TS Elementary Streams

The MPEG-2 transport stream protocol provides for a number of elementary media streams to be multiplexed together and carried as a unified synchronized package. This document limits the media types that can be de-multiplexed from a transport stream to MPEG-2 and H.264/AVC motion imagery and KLV-encoded metadata. While there is no reason that other media types multiplexed into a MPEG-2 TS stream, for example audio, cannot be similarly produced as an RTP stream, at this time only guidance for motion imagery and metadata is provided.

Component media streams within a MPEG-2 transport stream are synchronized together through the presentation time stamp (PTS) and decode time stamp (DTS) that accompanies each component (motion imagery, audio, metadata) packetized elementary stream (PES). The PTS indicates when the decoded data is to be presented, or displayed. The PTS thus provides one timing reference for a corresponding RTP media stream.

Guidance for mapping a video packetized stream to RTP is found in IETF RFC 2250 [9] with a mapping of the PTS to a RTP timestamp stated as:

> "Presentation Time Stamps (PTS) of 32 bits with accuracy of 90 kHz shall be carried in the fixed RTP header. All packets that make up a [video] frame shall have the same time stamp."

Note: metadata carried using the Synchronous Metadata Multiplex Method (see MISB ST 1402 [15]) is mapped in a similar fashion using its corresponding PTS.

When a transport stream contains two or more component media streams that are to be produced as two or more RTP streams, for example, a motion imagery elementary stream and a metadata elementary stream, the Real Time Control Protocol (RTCP) should be used to support synchronization between the two media streams. RTCP carries a common reference clock (wall clock) that is the reference clock for the time stamps in the component media streams, and thus provides synchronization of the component streams at a decoder.

Requirement	
ST 0804.4-19	When a MPEG-2 transport stream contains two or more component media streams that are to be produced as two or more RTP streams, the Real Time Control Protocol (RTCP) shall be used in accordance with IETF RFC 3350 [2].

11 Supporting Streaming Components

At a high level a motion imagery architecture consists of a data provider, or source of motion imagery, audio, metadata, etc. media components, and a data receiver of one or more of these same media components. These media components are typically synchronized to one another; that is, the events occurring within a media correlate directly with events in the corresponding media components. Lip sync is an example, where mouth movement corresponds to words spoken. RTCP (Real Time Control Protocol) [2] provides this synchronization between media components.

A second protocol called RTSP (Real Time Protocol) provides stream playback control for play, stop, rewind, and fast forward. Application needs will determine if RTCP or RTSP or both are warranted. Single media delivery with no control can use RTP alone. In these simple cases, the value in RTP is providing time stamp and packet count information useful in compensating for lost and re-ordered packets and network jitter.

11.1 Real Time Control Protocol (RTCP)

RTCP is a useful companion protocol that provides stream synchronization and bi-directional feedback between the sender and the receiver regarding the quality of a RTP session. As an example, RTCP allows a sender to provide a timing reference and a number of bytes and packets that have been sent. Conversely, it allows a receiver to inform a sender about the quantity of packets lost, and a measure of the packet arrival jitter. RTCP usually accompanies streams delivered using RTP, and thus industry equipment may not function without it.

RTP time stamps, present in the RTP header, represent the sampling instant of the first octet of data in a media frame (motion imagery frame, for example). IETF RFC 3550 describes how to synchronize content transported in RTP. A summary for the synchronization of motion imagery and metadata is given below:

1. The RTP time stamp is expressed in units of a clock, which is required to increase monotonically and linearly. The frequency of this clock is specified for each payload format, either explicitly or by default. Often, but not necessarily, this clock is the sampling clock.
2. RTCP data is carried in RTCP packets. There are five types of RTCP packets, one of which is the Sender Report (SR) RTCP packet type. Each RTCP SR packet contains a RTP time stamp and a NTP (Network Time Protocol) time stamp; both time stamps correspond to the same instant in time. However, the RTP time stamp is expressed in the same units as RTP time stamps in data packets, while the NTP time stamp is expressed in "wall clock" time; see clause 4 of RFC 3550.
3. Synchronized playback of streams is only possible if the streams use the same wall-clock to encode NTP values in SR packets. Receivers can achieve synchronization by using the correspondence between RTP and NTP time stamps. To synchronize a motion imagery stream and a metadata stream, one needs to receive an RTCP SR packet for the motion imagery stream, and an RTCP SR packet for the metadata stream. These SR packets provide a pair of NTP timestamps and their corresponding RTP timestamps that are used to align and synchronize the media.

The update rate of RTCP sender packets is typically 5 seconds, which means that upon entering a streaming session there may be an initial delay – on average a 2.5 second duration if the default RTCP timing rules are used – when the receiver does not yet have the necessary information to perform inter-stream synchronization.

Requirement	
ST 0804.4-20	When motion imagery and metadata are required to be time synchronized at the receiver a RTP server shall transmit RTCP sender reports (SR).

11.2 Control

The Real Time Streaming Protocol (RTSP) defined by IETF RFC 2326 [16] provides an application level protocol to interactively control the streaming of motion imagery.

RTSP provides a flexible protocol framework for controlling data streams with real-time properties. The following restrictions apply to ensure interoperability between endpoints supporting motion imagery data streams when this level of control is required or desired:

Requirement	
ST 0804.4-21	RTSP clients and servers shall implement all required features of the minimal RTSP implementation described in RFC 2326 [16] Appendix D.
ST 0804.4-22	RTSP clients and servers shall implement the PLAY method.
ST 0804.4-23	RTSP clients and servers shall support RTP/AVP transport in the “Transport” header.
ST 0804.4-24	RTSP servers shall send the “RTP-Info” header for unicast sessions.
ST 0804.4-25	RTSP servers and clients shall support aggregated control of presentations.
ST 0804.4-26	At most one RTSP session shall be “active” on a connection between an RTSP client and an RTSP server at any one time.

ST 0804.4-27	When the DESCRIBE method is implemented, a SDP shall be the description format as specified in RFC 2326[16] Appendix C.

Additional Recommendations

- When RTP/AVP transport is used for a unicast session, clients should include the “client_port” parameter in the “Transport” header and servers should include the “server_port”, “source”, and “ssrc” parameters in the “Transport” header.
- A RTSP session becomes “active” when it is first referenced in a “Session” header. A RTSP session is no longer “active” after a TEARDOWN request has been issued for that session.
- RTSP clients and servers should implement the DESCRIBE method.
- RTSP clients should generate the following RTSP headers when appropriate: “Bandwidth”, “Cache-Control”, “If-Modified-Since”, “User-Agent”. RTSP servers should correctly interpret these headers when present.
- RTSP servers should generate the following RTSP headers when appropriate: “Cache-Control”, “Expires”, “Last-Modified”, “Server”. RTSP clients should correctly interpret these headers when present.

11.3 Description and Addressing

Before a session can begin there needs to be setup information exchanged between sender and receiver; Session Description Protocol (SDP) [17] is one common protocol used. SDP provides a flexible text-based language for describing media streams and relating them temporally.

Requirement	
ST 0804.4-28	When a RTP server is required to describe a RTP multi-media session, the RTP server shall use SDP: Session Description Protocol, IETF RFC 4566[17] to describe the session.

When delivering SDP data via RTSP, it is recommended that the SDP data be formatted according to Appendix C of RTSP [16] to ensure optimal interoperability.

As a note, media description in SDP format can be delivered in several ways; examples include HTTP, RTSP, SMTP, SAP [19] and SIP.

12 Sample RTSP/RTP Session

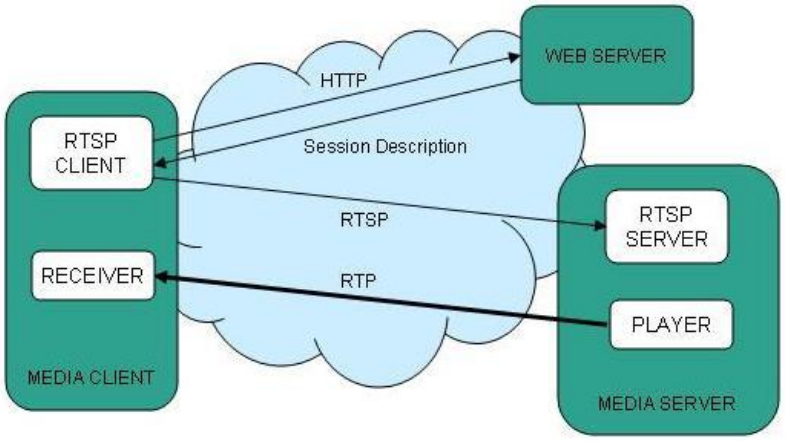

Figure 2: RTSP/RTP Server/Client Communication

The following commands, illustrated as state and requests in Figure 3, control a RTP session:

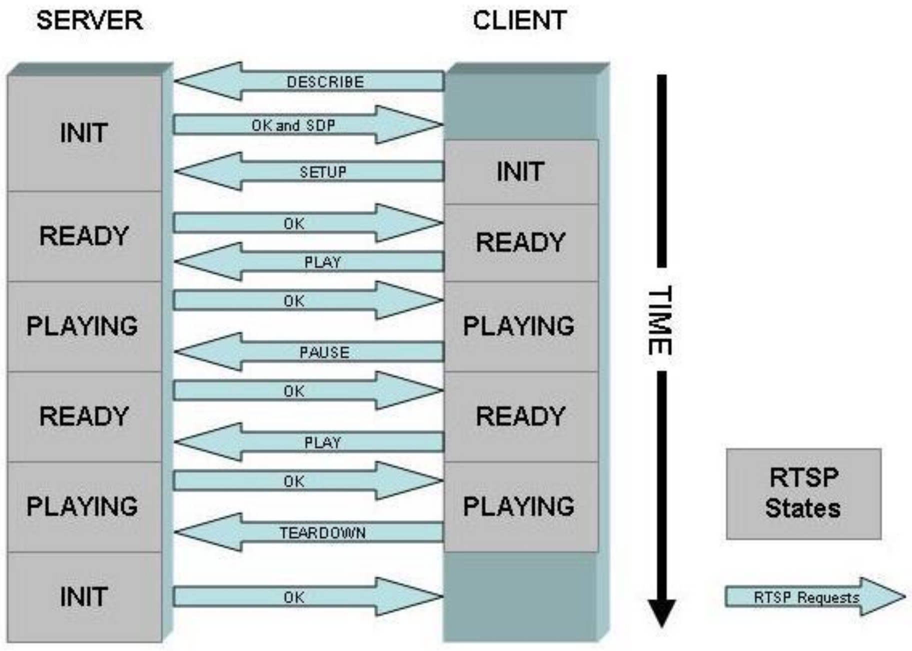

Figure 3: Server/Client RTSP Interaction

DESCRIBE: Issued by the client to retrieve a description of a presentation or media object on the server, corresponding to the *Universal Resource Locator* (URL) sent in the request. The response is typically in the form of the *Session Description Protocol* (SDP) and gives details such as the encoding types of the media, media duration, authors, etc. DESCRIBE allows clients to learn about a clip prior to streaming, and also to check if the client can support the media format.

OPTIONS: Informs the sender what other valid requests it may issue i.e. what requests are supported by the corresponding client or server for the specified content at a given time. Illegal requests by either the client or server can be avoided with this operation.

SETUP: Transport of the requested media is configured using this command. Details such as the transport protocol and the port numbers to use are submitted to the server, so the content is delivered in a manner appropriate for the client.

PLAY: Tells the server to start transmitting the requested media content as specified in the associated SETUP command. Unless a SETUP request has been issued and acknowledged, it is illegal to call the PLAY command. The absolute playback time and duration are also specified in this command, so operation similar to fast forward and rewind on a VCR can be achieved with this command if the media can support such functionality e.g. live motion imagery streams cannot be scanned forward.

PAUSE: Temporarily interrupts the delivery of the media currently playing in the session. PLAY must have been successfully requested in order to allow pausing of a motion imagery stream to a client. Resuming a paused media session is achieved using the PLAY request.

TEARDOWN: Terminates a stream or session. Once this command has been successfully issued the SETUP request must be called again before media content can be streamed again.

Other optional requests defined in the RTSP standard include ANNOUNCE, SET_PARAMETER, GET_PARAMETER, RECORD and REDIRECT. States for each session are maintained by the server to ensure that only valid requests are processed, and that an error response is replied to invalid requests. To aid the server in determining if a request is valid a number of possible server states are used:

1. **Init:** the initial state meaning that the server has received no valid SETUP request.
2. **Ready:** the previous SETUP request was successful and acknowledged and the server is waiting to start or finish playing, or a valid PAUSE request has been called.
3. **Playing:** a previous PLAY request was successful and content is currently being streamed to the client.

The example in Figure 3 illustrates an RTSP interaction between a client and server, highlighting the client and server states in response to different RTSP requests. In the session shown, the clients asks for a description of content contained on the server using the DESCRIBE command and the server delivers information in SDP format with relevant details of the media queried by the client. The client then issues a SETUP request, and the server configures the delivery of the stream to suit the clients preferred setup. PLAY is then issued by the client and the server begins transmitting the media data. A PAUSE request prompts the server to halt the stream temporarily; the server waits in the Ready state for the client to issue further instructions. When the client requests PLAY the stream resumes. Finally, the client issues a TEARDOWN request; the server terminates the session and returns to the Init state.

13 Appendix A RTP Payload Format for SMPTE ST 336 Encoded Data Implementation Guidance (Informative)

13.1 Background

This documents a proof-of-concept implementation based on the IETF RFC 6597 RTP Payload Format for SMPTE ST 336 Encoded Data [8], and is intended to serve as guidance for future "production class" implementations.

13.2 Definitions

KLVunit: a logical collection of all KLV items that are to be presented at a specific time. A KLVunit is comprised of one or more KLV items. Compound items (sets, packs) are allowed as per [SMPTE ST 336], but the contents of a compound item MUST NOT be split across two KLVunits. Multiple KLV items in a KLVunit occur one after another with no padding or stuffing between items.

13.3 Introduction

A client and a server implementation based on RFC 6597 [8] is described. To permit rapid prototype development, both implementations utilized and extended existing software applications. The guidance herein will be most directly applicable to implementers seeking to expand an existing RTP-capable software package with KLV/SMPTE ST 336 carriage capability in accordance with RFC 6597. However, completely new RTP implementations may also benefit.

For ease of session creation and association of multiple, related RTP sessions (motion imagery, audio, and KLV metadata) RTSP was used on both the client and server side of the implementation. RTSP provides a means for a client to connect to a server, ask for a description of available RTP-accessible media programs, and request that the server stream the media to the client over negotiated lower-level transport protocols (generally, UDP over IP).

It is assumed that the reader is familiar with general RTP concepts. Such background material can be found in [2], [3] and [21].

13.3.1 Client Implementation

13.3.1.1 Basis for Implementation

The client implementation was constructed by extending an existing software package already capable of initiating RTSP sessions, synchronizing, and playing back multiple RTP-based sessions. Additionally, the software package could support SMPTE ST 336 (KLV) parsing and interpretation of relevant KLV items from non-RTP sources (MPEG-2 TS files or network streams, generally).

Because of the existing RTSP and RTP stack on the client end, as well as the capable KLV parsing and interpretation, implementation work was isolated to two main components: 1) extending the RTSP session initiation to recognize the KLV payload format in session descriptions (SDP), and 2) handling the new KLV payload format at the individual RTP session level.

13.3.1.2 Extension of RTSP Session Initiation

The client application already contained a table of known payload format type descriptions, keyed on media type name (legal media type names can be found in the IANA RTP Parameters Registry, http://www.iana.org/assignments/rtp-parameters). This table is used during RTSP session initiation when parsing SDP descriptions of the RTP sessions available from the server.

Entries in the table map the media type name to several processes needed for that type of media such as:

- Procedures to parse parameters in the SDP description that further describe the media's format
- Procedures to handle RTP packet headers and payloads consistent with the media format
- Identifiers that indicate how the raw media data is processed once the RTP layer processing is completed

Expanding the RTSP session initiation code to handle the new KLV payload format involved simply adding a table entry for the corresponding media type described in RFC 6597. The entry provides the above three pieces of information as detailed below:

- **Parsing Payload Format Parameters:** The KLV payload format contains only one parameter, the clock rate. This is a very standard RTP parameter common to many payload formats. As such, the software already contained a re-usable routine to parse the rate parameter and subsequently use it to interpret RTP packet timestamps in the KLV-over-RTP session.
- **Procedures to handle RTP Packet Headers and Payloads:** Because of the similarities between the KLV/SMPTE ST 336 payload format specification and other popular payload formats, the implementation was able to re-use existing header and payload handling functions for the new KLV payload format. The next section describes the similarities and processing in further detail.
- **Identifiers Indicating How to Process Raw Media:** The software package already contained a KLV parser and processing framework; because of this, processing of KLVunits was simply mapped and routed to this pre-existing framework. See the next section for additional details.

13.3.1.3 Handling of the KLV Payload Format at the RTP Session Level

Once RTSP negotiation is complete, the individual RTP sessions are established. This section discusses the routines (mentioned briefly in the previous section) which handle the KLV/SMPTE ST 336 payload format.

RTP packet header and payload handling for the KLV payload format is based upon other popular video payload formats. A KLVunit is the same as a video frame in concept: it is all the data associated with a particular time, potentially spread across multiple RTP packet payloads for transport efficiency. The handling of the M-bit in the RTP header also parallels that of popular frame-based video formats. In addition, the KLV payload format does not define any additional RTP or payload headers. Because of the similarity with other popular video formats already supported by the software package, existing routines can be used without modification to provide proper RTP packet processing. The effective outputs of these routines are KLVunits together with their associated timestamps.

Once stripped of the RTP headers and concatenated as appropriate by the above-described handling mechanisms into whole KLVunits, processing of the KLV data is precisely equivalent to existing KLV processing and processing used with file or raw UDP network-based media. There is little concern about recovery from lost packets and data given the robustness of the existing KLV parser and processor. This downstream processor can deal with damaged KLVunits similarly to corrupt or malformed portions of file-based KLV streams. For implementations with error-intolerant KLV parsers, an additional step of discarding damaged KLVunits may be necessary to prevent problems in subsequent processing.

13.3.2 Server Implementation

13.3.2.1 Basis for Implementation

The server implementation was built by modifying the open source application "VLC" by the VideoLAN Project. VLC contains an RTP stack for streaming many popular video and audio formats from file-based media or re-streaming from other network sources. It also contains an RTSP server for session description and setup. This made it an ideal candidate for implementing the SMPTE ST 336/KLV payload format for RTP through extension.

To provide a concrete example for the community, as well as maintaining consistency with the open source philosophy that VLC is built upon, the actual source code modifications made to implement the KLV payload format for RTP is available for download from the MISB website.

13.3.2.2 Server Considerations

The server implementation was scoped to implement the KLV payload format for RTP described in RFC 6597 [8] as a primary goal. To facilitate this, a data source had to be considered to provide the data for streaming. VLC presently does not process KLV metadata, so consideration went into support of various source data formats to avoid turning the RTP-oriented proof-of-concept into the much larger project of generically supporting KLV streams in VLC.

Based on this, and the broad availability of test data, it was decided to support MPEG-2 transport stream sources (from any source, including file or network) containing asynchronous KLV in private data streams (in accordance with SMPTE RP 217 [22]) as the source for data to stream. VLC's transport stream de-multiplexer needed to be modified to recognize and support KLV data.

Although MPEG-2 transport streams containing synchronous metadata would have proved a more accurate data source, asynchronous data sources were chosen because of the wide-spread availability of data samples.

13.3.2.3 MPEG-2 Transport Stream De-multiplexer Modifications

As mentioned above, VLC does not contain any notion of KLV metadata processing. While a complete description of the modifications necessary to provide this support is out of scope for this document, a brief discussion is provided for the purpose of conveying background context to the available download of the overall KLV payload format for RTP server-side implementation.

Routines were added to detect the registration format descriptor ("KLVA") in private data stream descriptions that indicate a SMPTE RP217 compliant KLV private data stream.

VLC does not contain a track "category" for metadata. Rather than complicating this rapid-prototype project with adding a new track category, it was decided to use the subtitles track category. Any detected KLV streams were mapped to subtitle-category tracks with the FourCC codec code "KLVA".

Routines were also added to the transport stream de-multiplexer to synthesize timestamps (based on the best-approximation stream proximity) of the otherwise timestamp-less asynchronous KLV data. The timestamps are used for further downstream delivery timing, and eventually used to populate RTP packet header timestamps.

13.3.2.4 KLV Stream-Out Packetizer

VLC's main streaming module (regardless of streaming protocol) is the "stream output" module. Prior to media being streamed, it is sent through a "packetizer" module. The packetizer creates basic payload units used by the streaming modules.

In the KLV payload format, the basic payload unit is the "KLVunit". As described in the client implementation above, the KLVunit is conceptually equivalent to a "frame" for motion imagery in that it contains all the data relevant to a given instant in time. Thus, a packetizer for KLV was authored to collect and concatenate any disjoint KLV data with the same timestamp.

This module ensures that the RTP output module receives whole KLVunits for output processing, thus completely taking care of the preparation for insertion into RTP packet payloads.

13.3.2.5 RTSP SDP Output

An RTSP client requests a description of the available RTP sessions when it connects to an RTSP server. To properly describe an available KLV session in accordance with the SMPTE336M/KLV payload format specification, an addition to the SDP synthesis code was added to map VLC streams with an internal codec identifier of "KLVA" to a media type code of "application/smpte336m".

The only payload format parameter for the KLV payload format is the clock rate; because this is such a common parameter, VLC already defaults this appropriately for all RTP stream descriptions.

13.3.2.6 RTP Packet Header and Payload Creation

The KLV payload format is very similar in its use of the RTP header (timestamp, M-bit) to most popular motion imagery payload formats (this is described in further detail in the client implementation guidance above). Because of this similarity, and the fact that VLC already supports RTP streaming of many popular motion imagery formats, it was possible to reuse existing packet creation functionality.

The existing packet creation function, intended primarily to take motion imagery frames and create one or more RTP packets, could be directly applied to a KLVunit.

It is expected that most existing, robust RTP implementations can be extended to support the KLV payload format by reusing existing functionality in this way.